47. (Amended) The computer program product of claim 45, wherein one or more proxy support elements of the C++ primitive proxy class is an overloaded operator that permits instances the C++ primitive proxy class 52 to be used on the left-hand side of one or more syntactical productions.

90. (Amended) The method of claim 89, wherein:

act (a) includes determining that the type of the first component is [one of the following Java components: a Java interface, and] a Java class, and

act (b) includes[: (i) ] transforming the Java [component] class into a C++ proxy class.

91. (Amended) The method of claim 90, wherein act (b) includes:

generating, within the C++ proxy class, [includes] one or more proxy support elements.

92. (Amended) The method of claim 89, wherein:

act (a) includes determining that the type of the first component is a Java array, and

act (b) includes[: (i) ] transforming the Java array into a C++ proxy class.

93. (Amended The method of claim 92, wherein act (b) includes:

generating, within the C++ proxy class, [includes] one or more proxy support elements.

94. (Amended) The method of claim 89, wherein:

act (a) includes determining that the type of the first component is a Java method, and

act (b) includes[:(i) ] transforming the Java method into a C++ proxy method.

95. (Amended) The method of claim 89, wherein:

act (a) includes determining that the type of the first component is a Java field, and

act (b) includes[: (i) ] transforming the Java field into a C++ proxy field.

Please add claims 105-145 as follows:

105. The computer program product of claim 12, wherein at least one of the proxy support elements of the C++ proxy class allows usage of null in C++ corresponding to usage of a null Java object reference in Java.

106. The computer program product of claim 105, wherein the C++ proxy class includes a conversion constructor to create a stand-alone proxy instance of the C++ proxy class initialized to not refer to any Java instance.

107. The computer program product of claim 12, wherein at least one of the proxy support elements provides a semantic usability to the C++ proxy class that closely corresponds to the semantic usability of a Java cast expression corresponding to the Java component.

108. The computer program product of claim 107, wherein the C++ proxy class includes a static class method that provides the semantic usability to the C++ proxy class that closely corresponds to the semantic usability of a Java cast expression corresponding to the Java component.

109. The computer program product of claim 12, wherein at least one of the proxy support elements provides an ability to map an instance of the C++ proxy class to an object that represents the Java component.

110. The computer program product of claim 109, wherein the C++ proxy class includes a framework support method that provides the ability to map an instance of the C++ proxy class to an object that represents the Java component.

111. The method of claim 89, wherein act (b) includes:

(i) transforming the Java component into a C++ proxy class that includes one or more proxy support elements.

112. The method of claim 111, wherein one or more proxy support elements of the C++ proxy class allow an instance of the C++ proxy class to be context-aware.

113. The method of claim 112, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy instance field.

114. The method of claim 112, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy static field.

115. The method of claim 112, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy array element.

116 The method of claim 112, wherein one of the proxy support elements allows an instance of the C++ proxy class to be aware of being a C++ proxy stand-alone object.

117. The method of claim 112, wherein act (b)(i) includes:

generating a proxy layer, and wherein awareness of the contexts is required by the proxy layer.

118. The method of claim 117, wherein act (b)(i) includes:

coding the proxy layer using the Java Native Interface.

119. The method of claim 111, wherein at least one of the proxy support elements of the C++ proxy class allows usage of null in C++ corresponding to usage of a null Java object reference in Java.

120. The method of claim 119, wherein act (b)(i) includes:

generating a conversion constructor within the C++ proxy class, the conversion constructor for creating a stand-alone proxy instance of the C++ proxy class initialized to not refer to any Java instance.

121. The method of claim 111, wherein at least one of the proxy support elements provides a semantic usability to the C++ proxy class that closely corresponds to the semantic usability of a Java cast expression corresponding to the Java component.

122. The method of claim 121, wherein act (b)(i) includes:

generating a static class method within the C++ proxy class, the static class method providing the semantic usability to the C++ proxy class that closely corresponds to the semantic usability of a Java cast expression corresponding to the Java component.

123. The method of claim 111, wherein at least one of the proxy support elements provides an ability to map an instance of the C++ proxy class to an object that represents the Java component.

124. The method of claim 123, wherein act (b)(i) includes:

generating a framework support method within the C++ proxy class, the framework support method providing the ability to map an instance of the C++ proxy class to an object that represents the Java component.

125. The method of claim 91, wherein one or more of the proxy support elements allow an instance of the C++ proxy class to be context-aware.

126. The method of claim 90, wherein the Java class is declared abstract, and act (b) includes:

defining the C++ proxy component to be instantiable.

127. The method of claim 93, wherein one or more of the proxy support elements allow an instance of the proxy component to be context-aware.

128. The method of claim 92 , wherein the Java array has a length attribute, and act (b) includes:

defining the proxy class to include a length field corresponding to the length attribute of the Java array.

129. The method of claim 92, wherein the Java array has an element type, and act (b) includes:

defining the C++ proxy class to have an element type corresponding to the element type of the Java array; and

generating, within the C++ proxy class, one or more array subscript operators that return a context-aware instance of the proxy class's type.

130. The method of claim 92, wherein the Java array has a primitive element type, and act (b) includes:

defining the C++ proxy class to have a primitive element type corresponding to the Java array primitive element type; and

generating one or more array subscript operators that return a primitive value.

131. The method of claim 94, wherein act (b) includes:

defining the C++ proxy method to have a constness based on a mutability of the Java method.

132. The method of claim 94, wherein act (b) includes:

defining the C++ proxy method to declare a return type that has a mutability based on a mutability of a return type declared by the Java method.

133. The method of claim 94, wherein act (b) includes:

defining the C++ proxy method to pass one or more arguments; and

defining each argument to have a mutability based on a mutability of a corresponding argument passed by the Java method.

134. The method of claim 94, wherein act (b) includes:

defining the C++ proxy method to throw an exception based on the Java method being defined to throw an exception.

135. The method of claim 94, wherein act (b) includes:

defining the C++ proxy method not to throw an exception based on the Java method being defined to throw an exception.

136. The method of claim 94, wherein act (b) includes:

determining whether the C++ proxy method is declared virtual or not declared virtual from one or more of the following aspects of the Java method: polymorphicability, mutability, and instantiability; and

defining the C++ proxy method to be declared virtual or not declared virtual based on the determination.

137. The method of claim 95, wherein act (b) includes:

declaring the C++ proxy field to be of type C++ proxy class; and

generating, within the C++ proxy class, one or more proxy support elements that allow an instance of the C++ proxy class to be context-aware, such that an instance of the C++ proxy field is context-aware.

138. The method of claim 95, wherein the Java field is of primitive type, act (b) including:

defining the C++ proxy field to be declared of type C++ primitive proxy class.

139. The method of claim 138, wherein act (b) further includes:

declaring the C++ proxy field to be of type C++ proxy class; and

generating, within the C++ proxy class, one or more proxy support elements that allow an instance of the C++ proxy class to be context-aware, such that an instance of the C++ proxy field is context-aware.